Smaller Selection Networks for Cardinality Constraints Encoding

Michał Karpiński, Marek Piotrów

Institute of Computer Science, University of Wrocław
Joliot-Curie 15, 50-383 Wrocław, Poland
{karp,mpi}@cs.uni.wroc.pl

Abstract. Selection comparator networks have been studied for many years. Recently, they have been successfully applied to encode cardinality constraints for SAT-solvers. To decrease the size of generated formula there is a need for constructions of selection networks that can be efficiently generated and produce networks of small sizes for the practical range of their two parameters: n – the number of inputs (boolean variables) and k – the number of selected items (a cardinality bound). In this paper we give and analyze a new construction of smaller selection networks that are based on the pairwise selection networks introduced by Codish and Zanon-Ivry. We prove also that standard encodings of cardinality constraints with selection networks preserve arc-consistency.

1 Introduction

Comparator networks are probably the simplest data-oblivious model for sorting-related algorithms. The most popular construction is due to Batcher [3] and it's called *odd-even* sorting network. For all practical values, this is the best known sorting network. However, in 1992 Parberry [9] introduced the serious competitor to Batcher's construction, called *pairwise* sorting network. In context of sorting, pairwise network is not better than odd-even network, in fact it has been proven that they have exactly the same size and depth. As Parberry said himself: *"It is the first sorting network to be competitive with the odd-even sort for all values of n"*. There is a more sophisticated relation between both types of network and their close resemblance. For overview of sorting networks, see Knuth [7] or Parberry [8].

In recent years new applications for sorting networks have been found, for example in encoding of *pseudo boolean constraints* and *cardinality constraints* for SAT-solvers. Cardinality constraints take the form $x_1 + x_2 + \cdots + x_n \sim k$, where $x_1, x_2, \ldots, x_n$ are boolean variables, k is a natural number, and $\sim$ is a relation from the set $\{=, <, \leq, >, \geq\}$. Cardinality constraints are used in many applications, the significant one worth mentioning arise in SAT-solvers. Using cardinality constraints with cooperation of SAT-solvers we can handle many practical problems that are proven to be hard. Works of Asín *et al.* [1,2] describe how to use odd-even sorting network to encode cardinality constraints into boolean formulas. In [6] authors do the same with pseudo boolean constraints.

It has already been observed that using selection networks instead of sorting networks is more efficient for the encoding of cardinality constraints. Codish and Zazon-Ivry [4] introduced pairwise cardinality networks, which are networks derived from pairwise sorting networks that express cardinality constraints. Two years later, same authors [5] reformulated the definition of pairwise selection networks and proved that their sizes are never worse than the sizes of corresponding odd-even selection networks. To show the difference they plotted it for selected values of n and k.

In this paper we give a new construction of smaller selection networks that are based on the pairwise selection ones and we prove that the construction is correct. We estimate also the size of our networks and compute the difference in sizes between our selection networks and the corresponding pairwise ones. The difference can be as big as $n \log n/2$ for $k = n/2$. Finally, we analyze the standard 3(6)-clause encoding of a comparator and prove that such CNF encoding of any selection network preserves arc-consistency with respect to a corresponding cardinality constraint.

The rest of the paper is organized in the following way: in Section 2 we give definitions and notations used in this paper. In Section 3 we recall the definition of pairwise selection networks and define auxiliary bitonic selection networks that we will use to estimate the sizes of our networks. In Section 4 we present the construction of our selection networks and prove its correctness. In Section 5 we analyze the sizes of the networks and, finally, in Section 6 we examine the arc-consistency of selection networks.

2 Preliminaries

In this section we will introduce definitions and notations used in the rest of the paper.

Definition 1 (input sequence). *Input sequence of length n is a sequence of natural numbers $\bar{x} = \langle x_1, \ldots, x_n \rangle$, where $x_i \in \mathbb{N}$ (for all $i = 1..n$). We say that $\bar{x} \in \mathbb{N}^n$ is sorted if $x_i \geq x_{i+1}$ (for each $i = 1..n-1$). Given $\bar{x} = \langle x_1, \ldots, x_n \rangle$, $\bar{y} = \langle y_1, \ldots, y_n \rangle$ we define concatenation as $\bar{x} :: \bar{y} = \langle x_1, \ldots, x_n, y_1, \ldots, y_n \rangle$. We will use the following functions from $\mathbb{N}^n$ to $\mathbb{N}^{n/2}$:*

$$left(\bar{x}) = \langle x_1, \ldots, x_{n/2} \rangle, \qquad right(\bar{x}) = \langle x_{n/2+1}, \ldots, x_n \rangle$$

Let $n, m \in \mathbb{N}$. We define a relation '$\succeq$' on $\mathbb{N}^n \times \mathbb{N}^m$. Let $\bar{x} = \langle x_1, \ldots, x_n \rangle$ and $\bar{y} = \langle y_1, \ldots, y_m \rangle$, then:

$$\bar{x} \succeq \bar{y} \iff \forall_{i \in \{1,\ldots,n\}} \forall_{j \in \{1,\ldots,m\}} \; x_i \geq y_j$$

Definition 2 (comparator). *Let $\bar{x} \in \mathbb{N}^n$ and let $i, j \in \mathbb{N}$, where $1 \leq i < j \leq n$. A comparator is a function $c_{i,j}$ defined as:*

$$c_{i,j}(\bar{x}) = \bar{y} \iff y_i = \max\{x_i, x_j\} \wedge y_j = \min\{x_i, x_j\} \wedge \forall_{k \neq i,j} \; x_k = y_k$$

Definition 3 (comparator network). *We say that $f^n : \mathbb{N}^n \to \mathbb{N}^n$ is a comparator network of order n, if it can be represented as the composition of finite number of comparators, namely, $f^n = c_{i_1,j_1} \circ \cdots \circ c_{i_k,j_k}$. The size of comparator network (number of comparators) is denoted by $|f^n|$. Comparator network of size 0 is denoted by id^n.*

Traditionally comparator networks are presented as circuits that receives n inputs and permutate them using comparators connected by "wires". Each comparator has two inputs and two outputs. The "upper" output is the maximum of inputs, and "lower" one is minimum. As an example look at Figure 1, where we present a comparator network of order 4, max^4, that outputs maximum from 4 inputs on its first output, namely, $y_1 = \max\{x_1, x_2, x_3, x_4\}$. It is well known that $|max^n| = n - 1$. We will often omit explicit declaration of order of comparator network when it is not ambiguous.

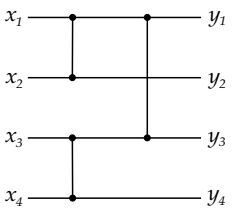


Fig. 1

Definition 4 (bitonic sequence). *A sequence $\bar{x} \in \mathbb{N}^n$ is a bitonic sequence if $x_1 \leq \ldots \leq x_i \geq \ldots \geq x_n$ for some i, where $1 \leq i \leq n$, or a circular shift of such sequence. We distinguish a special case of a bitonic sequence:*

- v-shaped*, if $x_1 \geq \ldots \geq x_i \leq \ldots \leq x_n$*

and among v-shaped sequences there are two special cases:

- nondecreasing*, if $x_1 \leq \ldots \leq x_n$,*
- nonincreasing*, if $x_1 \geq \ldots \geq x_n$.*

Definition 5 (sorting network). *A comparator network f^n is a* sorting network*, if for each $\bar{x} \in \mathbb{N}^n$, $f^n(\bar{x})$ is sorted.*

Two types of sorting networks are of interest to us: *odd-even* and *pairwise*. Based on their ideas, Knuth [7] (for odd-even network) and Codish and Zazon-Ivry [5] (for pairwise network) showed how to transform them into selection networks (we name them $oe\_sel^n_k$ and $pw\_sel^n_k$ respectively).

Definition 6 (top k sorted sequence). *A sequence $\bar{x} \in \mathbb{N}^n$ is top k sorted, with $k \leq n$, if $\langle x_1, \ldots, x_k \rangle$ is sorted and $\langle x_1, \ldots, x_k \rangle \succeq \langle x_{k+1}, \ldots, x_n \rangle$.*

Definition 7 (selection network). *A comparator network f^n_k (where $k \leq n$) is a* selection network*, if for each $\bar{x} \in \mathbb{N}^n$, $f^n_k(\bar{x})$ is top k sorted.*

To simplify the presentation we assume that n and k are powers of 2.

A clause is a disjunction of literals (boolean variables x or their negation $\neg x$). A CNF formula is a conjunction of one or more clauses.

A unit propagation (UP) is a process, that for given CNF formula, clauses are sought in which all literals but one are false (say l) and l is undefined (initially

only clauses of size one satisfy this condition). This literal l is set to true and the process is iterated until reaching a fix point.

Cardinality constraints are of the form $x_1 + \ldots + x_n \sim k$, where $k \in \mathbb{N}$ and $\sim$ belongs to $\{<, \leq, =, \geq, >\}$. We will focus on cardinality constraints with less-than relation, i.e. $x_1 + \ldots + x_n < k$. An encoding (a CNF formula) of such constraint preserves arc-consistency, if as soon as $k-1$ variables among the x_i's become true, the unit propagation sets all other x_i's to false.

In [6] authors are using sorting networks for an encoding of cardinality constraints, where inputs and outputs of a comparator are boolean variables and comparators are encoded as a CNF formula. In addition, the k-th greatest output variable y_k of the network is forced to be 0 by adding $\neg y_k$ as a clause to the formula that encodes $x_1 + \ldots + x_n < k$. They showed that the encoding preserves arc-consistency.

A single comparator can be translated to a CNF formula in the following way: let a and b be variables denoting upper and lower inputs of the comparator, and c and d be variables denoting upper and lower outputs of a comparator, then:

$$fcomp(a, b, c, d) \Leftrightarrow (c \Leftrightarrow a \vee b) \wedge (d \Leftrightarrow a \wedge b)$$

is the *full encoding* of a comparator. Notice that it consists of 6 clauses. Let f be a comparator network. Full encoding ϕ of f is a conjunction of full encoding of every comparator of f.

In [2] authors observe that in case of $\sim$ being $<$ or $\leq$, it is sufficient to use only 3 clauses for a single comparator, namely:

$$hcomp(a, b, c, d) \Leftrightarrow \underbrace{(a \Rightarrow c)}_{(c1)} \wedge \underbrace{(b \Rightarrow c)}_{(c2)} \wedge \underbrace{(a \wedge b \Rightarrow d)}_{(c3)} \tag{1}$$

We call it: *half encoding*. In [2] it is used to translate odd-even sorting network to encoding that preserves arc-consistency. We show a more general result (with respect to both [6] and [2]), that half encoding of any selection network preserves arc-consistency for the "$<$" and "$\leq$" relations. Similar results can be proved for the "$=$" relation using the full encoding of comparators and for the "$>$" or "$\geq$" relations using an encoding symmetric to $hcomp(a, b, c, d)$, namely: $(d \Rightarrow a) \wedge (d \Rightarrow b) \wedge (c \Rightarrow a \vee b)$.

3 Pairwise and bitonic selection networks

Now we present two constructions for selection networks. First, we recall the definition of pairwise selection networks by Codish and Zazon-Ivry [5]. Secondly, we give the auxiliary construction of a *bitonic* selection network $bit\_sel_k^n$, that we will use to estimate the sizes of our improved pairwise selection network in Section 5.

Definition 8 (domination). *$\bar{x} \in \mathbb{N}^n$ dominates $\bar{y} \in \mathbb{N}^n$ if $x_i \geq y_i$ (for $i = 1..n$).*

Definition 9 (splitter). *A comparator network f^n is a* splitter *if for any sequence $\bar{x} \in \mathbb{N}^n$, if $\bar{y} = f^n(\bar{x})$, then $left(\bar{y})$ dominates $right(\bar{y})$.*

Observation 1. *We can construct splitter $split^n$ by joining inputs $\langle i, n/2+i\rangle$, for $i = 1..n/2$, with a comparator. Size of a splitter is $|split^n| = n/2$.*

Lemma 1. *If $\bar{b} \in \mathbb{N}^n$ is bitonic and $\bar{y} = split^n(\bar{b})$, then $left(\bar{y})$ and $right(\bar{y})$ are bitonic and $left(\bar{y}) \succeq right(\bar{y})$.*

Proof. See Appendix B of [3]. □

Network 1 ($pw\_sel_k^n$; see [5])**.** *Input: any $\bar{x} \in \mathbb{N}^n$.*

1. *If $k = 1$, return $max^n(\bar{x})$.*
2. *If $k = n$, return $oe\_sort^n(\bar{x})$.*
3. *Compute $\bar{y} = split(\bar{x})$.*
4. *Compute $\bar{l} = pw\_sel_k^{n/2}(\bar{y})$ and $\bar{r} = pw\_sel_{k/2}^{n/2}(\bar{y})$.*
5. *Compute $pw\_merge_k^n(\bar{l} :: \bar{r})$.*

Notice that since we introduced a splitter as the third step, in the recursive calls we need to select k top elements from the first half of $\bar{y}$, but only $k/2$ elements from the second half. The reason: $r_{k/2+1}$ cannot be one of the first k largest elements of $\bar{l} :: \bar{r}$. First, $r_{k/2+1}$ is smaller than any one of $\langle r_1, \ldots, r_{k/2}\rangle$ (by the definition of top k sorted sequence), and second, $\langle l_1, \ldots, l_{k/2}\rangle$ dominates $\langle r_1, \ldots, r_{k/2}\rangle$, so $r_{k/2+1}$ is smaller than any one of $\langle l_1, \ldots, l_{k/2}\rangle$. From this argument we make the following observation:

Observation 2. *If $\bar{l} \in \mathbb{N}^{n/2}$ is top k sorted, $\bar{r} \in \mathbb{N}^{n/2}$ is top $k/2$ sorted and $\langle l_1, \ldots, l_{k/2}\rangle$ dominates $\langle r_1, \ldots, r_{k/2}\rangle$, then k largest elements of $\bar{l} :: \bar{r}$ are in $\langle l_1, \ldots, l_k\rangle :: \langle r_1, \ldots, r_{k/2}\rangle$.*

The last step of Network 1 merges k top elements from $\bar{l}$ and $k/2$ top elements from $\bar{r}$ with so called *pairwise merger*. We will omit the construction of this merger, because it is not relevant to our work. We would only like to note, that its size is: $|pw\_merge_k^n| = k \log k - k + 1$. Construction of the merger as well as the detailed proof of correctness of network $pw\_sel_k^n$ can be found in Section 6 of [5].

Definition 10 (bitonic splitter). *A comparator network f^n is a bitonic splitter if for any two sorted sequences $\bar{x}, \bar{y} \in \mathbb{N}^{n/2}$, if $\bar{z} = bit\_split^n(\bar{x} :: \bar{y})$, then (1) $left(\bar{z}) \succeq right(\bar{z})$ and (2) $left(\bar{z})$ and $right(\bar{z})$ are bitonic.*

Observation 3. *We can construct bitonic splitter bit_split^n by joining inputs $\langle i, n-i+1\rangle$, for $i = 1..n/2$, with a comparator. Size of a bitonic splitter is $|bit\_split^n| = n/2$.*

We now present the procedure for construction of the bitonic selection network. We use the odd-even sorting network *oe_sort* and the network *bit_merge* (also by Batcher [3]) for sorting bitonic sequences as black-boxes. As a reminder: bit_merge^n consists of two steps, first we use $\bar{y} = split^n(\bar{x})$, then recursively compute $bit\_merge^{n/2}$ for $left(\bar{y})$ and $right(\bar{y})$ (base case, $n = 2$, consists of a single comparator). Size of this network is: $|bit\_merge^n| = n \log n/2$.

Bitonic selection network $bit\_sel^n_k$ is constructed by the following procedure.

Network 2 ($bit\_sel^n_k$)**.** *Input: any* $\bar{x} \in \mathbb{N}^n$*.*

1. *Let* $l = n/k$*. Partition input* $\bar{x}$ *into* l *consecutive blocks, each of size* k*, then sort each block with* oe_sort^k*, obtaining* $B_1, \ldots, B_l$*.*
2. *While* $l > 1$*, do the following:*
   (a) *Collect blocks into pairs* $\langle B_1, B_2\rangle, \ldots, \langle B_{l-1}, B_l\rangle$*.*
   (b) *Compute* $\bar{y_i} = bit\_split^{2k}(B_i :: B_{i+1})$ *for each* $i \in \{1, 3, \ldots, l-1\}$*.*
   (c) *Compute* $B'_{\lceil i/2\rceil} = bit\_merge^k(left(\bar{y_i}))$ *for each result of previous step.*
   (d) *Let* $l = l/2$*. Relabel* B'_i *to* B_i*, for* $1 \le i \le l$*.*

Theorem 1. *A comparator network* $bit\_sel^n_k$ *constructed by the procedure Network 2 is a selection network.*

Proof. Let $\bar{x} \in \mathbb{N}^n$ be the input to $bit\_sel^n_k$. After step one we get sorted sequences $B_1, \ldots, B_l$, where $l = n/k$. Let l_m be the value of l after m iterations. Let $B^m_1, \ldots, B^m_{l_m}$ be the blocks after m iterations. We will prove by induction that:

$P(m)$: if $B_1, \ldots, B_l$ are sorted and are containing k largest elements of $\bar{x}$, then after m-th iteration of the second step: $l_m = l/2^m$, $B^m_1, \ldots, B^m_{l_m}$ are sorted and are containing k largest elements of $\bar{x}$.

If $m = 0$, then $l = 1$, so $P(m)$ holds. We show that $\forall_{m\ge 0}\ (P(m) \Rightarrow P(m+1))$. Consider $(m+1)$-th iteration of step two. By the induction hypothesis $l_m = l/2^m$, $B^m_1, \ldots, B^m_{l_m}$ are sorted and are containing k largest elements of $\bar{x}$. We will show that $(m+1)$-th iteration does not remove any element from k largest elements of $\bar{x}$. To see this, notice that if $\bar{y_i} = bit\_split^{2k}(B^m_i :: B^m_{i+1})$ (for $i \in \{1, 3, \ldots, l_m - 1\}$), then $left(\bar{y_i}) \succeq right(\bar{y_i})$ and that $left(\bar{y_i})$ is bitonic (by Definition 10). Because of those two facts, $right(\bar{y_i})$ is discarded and $left(\bar{y_i})$ is sorted using bit_merge^k. After this, $l_{m+1} = l_m/2 = l/2^{m+1}$ and blocks $B^{m+1}_1, \ldots, B^{m+1}_{l_{m+1}}$ are sorted. Thus $P(m+1)$ is true.

Since $l = n/k$, then by $P(m)$ we see that the second step will terminate after $m = \log \frac{n}{k}$ iterations and that B_1 is sorted and contains k largest elements of $\bar{x}$. □

Schema of construction of bitonic selection network is shown in Figure 2. The size of bitonic selection network is:

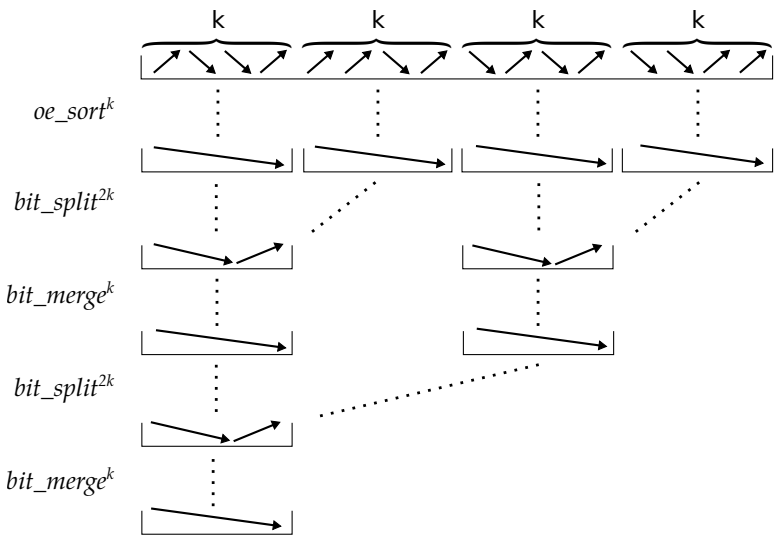


Fig. 2: Bitonic selection network – schema of construction

$$
\begin{aligned}
|bit\_sel_k^n| &= \frac{n}{k}|oe\_sort^k| + \left(\frac{n}{k} - 1\right)\left(|bit\_split^{2k}| + |bit\_merge^k|\right) \\
&= \frac{1}{4}n\log^2 k + \frac{1}{4}n\log k + 2n - \frac{1}{2}k\log k - k - \frac{n}{k} \qquad (2)
\end{aligned}
$$

In Figure 3 we present bitonic and pairwise selection networks for $n = 8$ and $k = 2$.

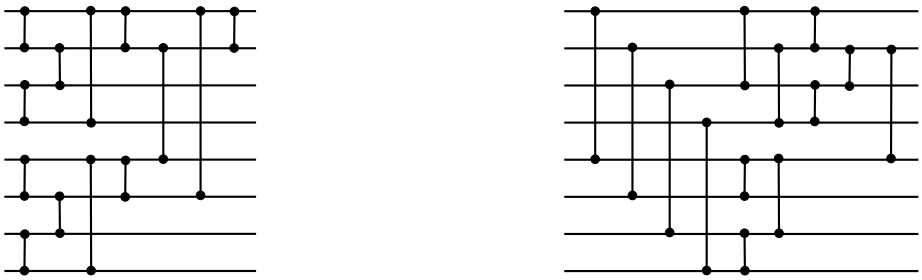


Fig. 3: a) bitonic selection network; b) pairwise selection network; $n = 8$, $k = 2$.

4 New Smaller Selection Networks

As mentioned in the previous section, only the first $k/2$ elements from the second half of the input are relevant when we get to the merging step in $pw\_sel_k^n$. We will exploit this fact to create a new, smaller merger. We will use the concept of bitonic sequences, therefore the new merger will be called $pw\_bit\_merge_k^n$ and the new selection network: $pw\_bit\_sel_k^n$. The network $pw\_bit\_sel_k^n$ is generated by substituting the last step of $pw\_sel_k^n$ with $pw\_bit\_merge_k^n$. The new merger consists of two steps:

Network 3 ($pw\_bit\_merge_k^n$)**.** *Input:* $\bar{l} :: \bar{r}$*, where* $\bar{l} \in \mathbb{N}^{n/2}$ *is top* k *sorted and* $\bar{r} \in \mathbb{N}^{n/2}$ *is top* $k/2$ *sorted and* $\langle l_1, \ldots, l_{k/2} \rangle$ *dominates* $\langle r_1, \ldots, r_{k/2} \rangle$*.*

1. *Compute* $\bar{y} = bit\_split^k(l_{k/2+1}, \ldots, l_k, r_1, \ldots, r_{k/2})$*, let* $\bar{b} = \langle l_1, \ldots, l_{k/2} \rangle :: \langle y_1, \ldots, y_{k/2} \rangle$*.*
2. *Compute* $bit\_merge^k(\bar{b})$*.*

Theorem 2. *The output of Network 3 consists of sorted* k *largest elements from input* $\bar{l} :: \bar{r}$*, assuming that* $\bar{l} \in \mathbb{N}^{n/2}$ *is top* k *sorted and* $\bar{r} \in \mathbb{N}^{n/2}$ *is top* $k/2$ *sorted and* $\langle l_1, \ldots, l_{k/2} \rangle$ *dominates* $\langle r_1, \ldots, r_{k/2} \rangle$*.*

Proof. We have to prove two things: (1) $\bar{b}$ is bitonic and (2) $\bar{b}$ consists of k largest elements from $\bar{l} :: \bar{r}$.

(1) Let j be the last index in the sequence $\langle k/2+1, \ldots, k \rangle$, for which $l_j > r_{k-j+1}$. If such j does not exist, then $\langle y_1, \ldots, y_{k/2} \rangle$ is nondecreasing, hence $\bar{b}$ is bitonic (nondecreasing). Assume that j exists, then $\langle y_{j-k/2+1}, \ldots, y_{k/2} \rangle$ is nondecreasing and $\langle y_1, \ldots, y_{k-j} \rangle$ is nonincreasing. Adding the fact that $l_{k/2} \geq l_{k/2+1} = y_1$ proves, that $\bar{b}$ is bitonic (v-shaped).

(2) By Observation 2, it is sufficient to prove that $\bar{b} \succeq \langle y_{k/2+1}, \ldots, y_k \rangle$. Since $\forall_{k/2<j\leq k}\ l_{k/2} \geq l_j \geq \min\{l_j, r_{k-j+1}\} = y_{3k/2-j+1}$, then $\langle l_1, \ldots, l_{k/2} \rangle \succeq \langle y_{k/2+1}, \ldots, y_k \rangle$ and by Definition 10: $\langle y_1, \ldots, y_{k/2} \rangle \succeq \langle y_{k/2+1}, \ldots, y_k \rangle$. Therefore $\bar{b}$ consists of k largest elements from $\bar{l} :: \bar{r}$.

The bitonic merger in step 2 receives a bitonic sequence, so it outputs a sorted sequence, which completes the proof. □

The first step of improved pairwise merger is illustrated in Figure 4. We use $k/2$ comparators in the first step and $k \log k/2$ comparators in the second step. We get a merger of size $k \log k/2 + k/2$, which is better than the previous approach. In the following it is shown that we can do even better and eliminate $k/2$ term.

The main observation is that the result of the first step of pw_bit_merge operation: $\langle b_1, b_2, \ldots, b_k \rangle$ is not only bitonic, but what we call *v-shape s-dominating.*

Definition 11 (s-domination). *A sequence* $\bar{b} = \langle b_1, b_2, \ldots, b_k \rangle$ *is s-dominating if* $\forall_{1\leq j\leq k/2}\ b_j \geq b_{k-j+1}$*.*

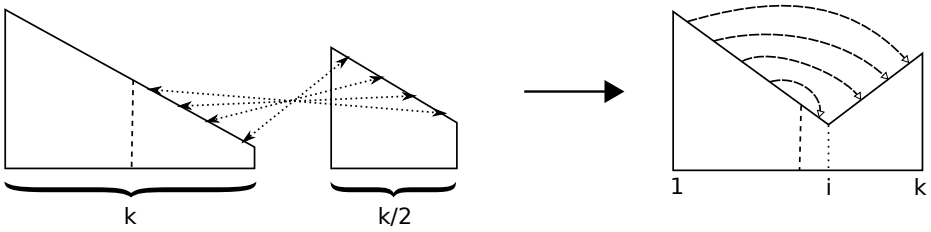


Fig. 4: Making the bitonic sequence. Arrows on the right picture show directions of inequalities. Sequence on the right is v-shape s-dominating at point i.

Lemma 2. *If $\bar{b} = \langle b_1, b_2, \ldots, b_k \rangle$ is v-shaped and s-dominating, then $\bar{b}$ is nonincreasing or $\exists_{k/2<i<k}\ b_i < b_{i+1}$.*

Proof. Assume that $\bar{b}$ is not nonincreasing. Then $\exists_{1\leq j<k}\ b_j < b_{j+1}$. Assume that $j \leq k/2$. Since $\bar{b}$ is v-shaped, b_{j+1} must be in nondecreasing part of $\bar{b}$. If follows that $b_j < b_{j+1} \leq \ldots \leq b_{k/2} \leq \ldots \leq b_{k-j+1}$. That means that $b_j < b_{k-j+1}$. On the other hand, $\bar{b}$ is s-dominating, thus $b_j \geq b_{k-j+1}$ – a contradiction. □

We will say that a sequence $\bar{b}$ is *v-shape s-dominating at point i* if i is the smallest index greater than $k/2$ such that $b_i < b_{i+1}$ or $i = k$ for a nonincreasing sequence.

Lemma 3. *Let $\bar{b} = \langle b_1, b_2, \ldots, b_k \rangle$ be v-shape s-dominating at point i, then $\langle b_1, \ldots, b_{k/4} \rangle \succeq \langle b_{k/2+1}, \ldots, b_{3k/4} \rangle$.*

Proof. If $\bar{b}$ is nonincreasing, then the lemma holds. From Lemma 2: $k/2 < i < k$. If $i > 3k/4$, then by Definition 4: $b_1 \geq \ldots \geq b_{3k/4} \geq \ldots \geq b_i$, so lemma holds. If $k/2 < i \leq 3k/4$, then by Definition 4: $b_1 \geq \ldots \geq b_i$, so $\langle b_1, \ldots, b_{k/4} \rangle \succeq \langle b_{k/2+1}, \ldots, b_i \rangle$. Since $b_i < b_{i+1} \leq \ldots \leq b_{3k/4}$, it suffices to prove that $b_{k/4} \geq b_{3k/4}$. By Definition 11 and 4: $b_{k/4} \geq b_{3k/4+1} \geq b_{3k/4}$. □

Definition 12 (half splitter). *A* half splitter *is a comparator network constructed by comparing inputs $\langle k/4+1, 3k/4+1 \rangle, \ldots, \langle k/2, k \rangle$ (normal splitter with first $k/4$ comparators removed). We will call it $half\_split^k$.*

Lemma 4. *If $\bar{b}$ is v-shape s-dominating, then $half\_split^k(\bar{b}) = split^k(\bar{b})$.*

Proof. Directly from Lemma 3. □

Lemma 5. *Let $\bar{b}$ be v-shape s-dominating. Following statements are true: (1) $left(half\_split^k(\bar{b}))$ is v-shape s-dominating; (2) $right(half\_split^k(\bar{b}))$ is bitonic; (3) $left(half\_split^k(\bar{b})) \succeq right(half\_split^k(\bar{b}))$.*

Proof. (1) Let $\bar{y} = left(half\_split^k(\bar{b}))$. First we show that $\bar{y}$ is v-shaped. If $\bar{y}$ is nonincreasing, then it is v-shaped. Otherwise, let j be the first index from the range $\{1, \ldots, k/2\}$, where $y_{j-1} < y_j$. Since $y_j = \max\{b_j, b_{j+k/2}\}$ and $y_{j-1} \geq$

$b_{j-1} \geq b_j$, thus $b_j < b_{j+k/2}$. Since $\bar{b}$ is v-shaped, element $b_{j+k/2}$ must be in nondecreasing part of $\bar{b}$. It follows that $b_j \geq \ldots \geq b_{k/2}$ and $b_{j+k/2} \leq \ldots \leq b_k$. From this we can see that $\forall_{j \leq j' \leq k/2}\ y_{j'} = \max\{b_{j'}, b_{j'+k/2}\} = b_{j'+k/2}$, so $y_j \leq \ldots \leq y_{k/2}$. Therefore $\bar{y}$ is v-shaped.

Next we show that $\bar{y}$ is s-dominating. Consider any j, where $1 \leq j \leq k/4$. By Definition 4 and 11: $b_j \geq b_{k/2-j+1}$ and $b_j \geq b_{k-j+1}$, therefore $y_j = b_j \geq \max\{b_{k/2-j+1}, b_{k-j+1}\} = y_{k/2-j+1}$, thus proving that $\bar{y}$ is s-dominating. Concluding: $\bar{y}$ is v-shape s-dominating.

(2) Let $\bar{z} = right(half\_split^k(\bar{b}))$. By Lemma 4: $\bar{z} = right(split^k(\bar{b}))$. We know that $\bar{b}$ is a special case of bitonic sequence, therefore using Lemma 1 we get that $\bar{z}$ is bitonic.

(3) Let $\bar{w} = half\_split^k(\bar{b})$. By Lemma 4: $\bar{w} = split^k(\bar{b})$. We know that $\bar{b}$ is a special case of bitonic sequence, therefore using Lemma 1 we get $left(\bar{w}) \succeq right(\bar{w})$. □

Using *half_split* and Batcher's *bit_merge* and successively applying Lemma 5 to the resulting v-shape s-dominating half of the output, we have all the tools needed to construct the improved pairwise merger using half splitters:

Network 4 ($pw\_hbit\_merge_k^n$)**.** *Input:* $\bar{l} :: \bar{r}$*, where* $\bar{l} \in \mathbb{N}^{n/2}$ *is top* k *sorted and* $\bar{r} \in \mathbb{N}^{n/2}$ *is top* $k/2$ *sorted and* $\langle l_1, \ldots, l_{k/2} \rangle$ *dominates* $\langle r_1, \ldots, r_{k/2} \rangle$*.*

1. *Compute* $\bar{y} = bit\_split^k(l_{k/2+1}, \ldots, l_k, r_1, \ldots, r_{k/2})$*, let* $\bar{b} = \langle l_1, \ldots, l_{k/2} \rangle :: \langle y_1, \ldots, y_{k/2} \rangle$*.*
2. *Compute* $half\_bit\_merge^k(\bar{b})$*:*
   (a) *If* $k = 2$*, return.*
   (b) *Let* $\bar{b'} = half\_split(b_1, \ldots, b_k)$*.*
   (c) *Recursively compute* $\bar{l'} = half\_bit\_merge^{k/2}(left(\bar{b'}))$*.*
   (d) *Compute* $\bar{r'} = bit\_merge^{k/2}(right(\bar{b'}))$*.*
   (e) *Return* $\bar{l'} :: \bar{r'}$*.*

The following theorem states that the construction of $pw\_hbit\_merge_k^n$ is correct.

Theorem 3. *The output of Network 4 consists of sorted* k *largest elements from input* $\bar{l} :: \bar{r}$*, assuming that* $\bar{l} \in \mathbb{N}^{n/2}$ *is top* k *sorted and* $\bar{r} \in \mathbb{N}^{n/2}$ *is top* $k/2$ *sorted and* $\langle l_1, \ldots, l_{k/2} \rangle$ *dominates* $\langle r_1, \ldots, r_{k/2} \rangle$*. Also* $|pw\_hbit\_merge_k^n| = k \log k/2$*.*

Proof. Since step 1 in Network 4 is the same as in Network 3, we can reuse the proof of Theorem 2 to deduce, that $\bar{b}$ is v-shaped and is containing k largest elements from $\bar{l} :: \bar{r}$. Also, since $\forall_{1 \leq j \leq k/2}\ l_j \geq l_{k-j+1}$ and $l_j \geq r_j$, then $b_j = l_j \geq \max\{l_{k-j+1}, r_j\} = b_{k-j+1}$, so $\bar{b}$ is s-dominating.

We prove by the induction on k, that if $\bar{b}$ is v-shape s-dominating, then the sequence $half\_bit\_merge^k(\bar{b})$ is sorted. For the base case, consider $k = 2$ and a v-shape s-dominating sequence $\langle b_1, b_2 \rangle$. By Definition 11 this sequence is already sorted and we are done. For the induction step, consider $\bar{b'} = half\_split^k(\bar{b})$. By Lemma 5 we get that $left(\bar{b'})$ is v-shape s-dominating and $right(\bar{b'})$ is bitonic.

Using the induction hypothesis we sort $left(\bar{b'})$ and using bitonic merger we sort $right(\bar{b'})$. By Lemma 5: $left(\bar{b'}) \succeq right(\bar{b'})$, which completes the proof of correctness.

As mentioned in Definition 12: $half\_split^k$ is just $split^k$ with the first $k/4$ comparators removed. So $half\_bit\_merge^k$ is just bit_merge^k with some of the comparators removed. Let's count them: in each level of recursion step we take half of comparators from $split^k$ and additional one comparator from the base case ($k = 2$). We sum them together to get:

$$1 + \sum_{i=0}^{\log k-2} \frac{k}{2^{i+2}} = 1 + \frac{k}{4}\left(\sum_{i=0}^{\log k-1} \left(\frac{1}{2}\right)^i - \frac{2}{k}\right) = 1 + \frac{k}{4}\left(2 - \frac{2}{k} - \frac{2}{k}\right) = \frac{k}{2}$$

Therefore we have:

$$|pw\_hbit\_merge_k^n| = k/2 + k\log k/2 - k/2 = k\log k/2$$

□

The only difference between pw_sel and our pw_hbit_sel is the use of improved merger pw_hbit_merge rather than pw_merge. By Theorem 3, we conclude that $|pw\_merge_k^n| \geq |pw\_hbit\_merge_k^n|$, so it follows that:

Remark 1. $|pw\_hbit\_sel_k^n| \leq |pw\_sel_k^n|$

5 Sizes of new selection networks

In this section we estimate the size of $pw\_hbit\_sel_k^n$. To this end we show that the size of $pw\_hbit\_sel_k^n$ is upper-bounded by the size of $bit\_sel_k^n$ and use this fact in our estimation. We also compute the exact difference between sizes of $pw\_sel_k^n$ and $pw\_hbit\_sel_k^n$ and show that it can be as big as $n\log n/2$. Finally we show graphically how much smaller is our selection network on practical values of n and k.

We have the recursive formula for the number of comparators of $pw\_hbit\_sel_k^n$:

$$|pw\_hbit\_sel_k^n| = \begin{cases} |pw\_hbit\_sel_k^{n/2}| + |pw\_hbit\_sel_{k/2}^{n/2}| + \\ +|split^n| + |pw\_hbit\_merge^k| & \text{if } k < n \\ |oe\_sort^k| & \text{if } k = n \\ |max^n| & \text{if } k = 1 \end{cases} \quad (3)$$

Lemma 6. $|pw\_hbit\_sel_k^n| \leq |bit\_sel_k^n|$.

Proof. Let $aux\_sel_k^n$ be the comparator network that is generated by substituting recursive calls in $pw\_hbit\_sel_k^n$ by calls to $bit\_sel_k^n$. Size of this network (for $1 < k < n$) is:

$$|aux\_sel_k^n| = |bit\_sel_k^{n/2}| + |bit\_sel_{k/2}^{n/2}| + |split^n| + |pw\_hbit\_merge^k| \quad (4)$$

Lemma 6 follows from Lemma 7 and Lemma 8 below, where we show that:

$$|pw\_hbit\_sel_k^n| \leq |aux\_sel_k^n| \leq |bit\_sel_k^n|$$

□

Lemma 7. *For $1 < k < n$ (both powers of 2), $|aux\_sel_k^n| \leq |bit\_sel_k^n|$.*

Proof. We compute both values from equations 2 and 4:

$$|aux\_sel_k^n| = \frac{1}{4}n\log^2 k + \frac{5}{2}n - \frac{1}{4}k\log k - \frac{5}{4}k - \frac{3n}{2k}$$
$$|bit\_sel_k^n| = \frac{1}{4}n\log^2 k + \frac{1}{4}n\log k + 2n - \frac{1}{2}k\log k - k - \frac{n}{k}$$

We simplify both sides to get the following inequality:

$$n - \frac{1}{2}k - \frac{n}{k} \leq \frac{1}{2}(n-k)\log k$$

which can be easily proved by induction. □

Lemma 8. *For $1 \leq k < n$ (both powers of 2), $|pw\_hbit\_sel_k^n| \leq |aux\_sel_k^n|$.*

Proof. By induction. For the base case, consider $1 = k < n$. If follows by definitions that $|pw\_hbit\_sel_k^n| = |aux\_sel_k^n| = n - 1$. For the induction step assume that for each $(n', k') \prec (n, k)$ (in lexicographical order) the lemma holds, we get:

$$\begin{aligned}
&|pw_hbit_sel_k^n| \\
&= |pw_hbit_sel_{k/2}^{n/2}| + |pw_hbit_sel_k^{n/2}| + |split^n| + |pw_hbit_merge^k| \\
&\qquad \textbf{(by the definition of } pw_hbit_sel\textbf{)} \\
&\leq |aux_sel_{k/2}^{n/2}| + |aux_sel_k^{n/2}| + |split^n| + |pw_hbit_merge^k| \\
&\qquad \textbf{(by the induction hypothesis)} \\
&\leq |bit_sel_{k/2}^{n/2}| + |bit_sel_k^{n/2}| + |split^n| + |pw_hbit_merge^k| \\
&\qquad \textbf{(by Lemma 7)} \\
&= |aux_sel_k^n| \\
&\qquad \textbf{(by the definition of } aux_sel\textbf{)}
\end{aligned}$$

□

Let $N = 2^n$ and $K = 2^k$. We will compute upper bound for $P(n,k) = |pw\_hbit\_sel_K^N|$ using $B(n,k) = |bit\_sel_K^N|$.

Lemma 9. *Let:*

$$P(n,k,m) = \sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}\left((k-j)2^{k-j-1}+2^{n-i-1}\right) + \sum_{i=0}^{m}\binom{m}{i}P(n-m,k-i).$$

Then $\forall_{0\le m\le \min(k,n-k)}\ P(n,k,m) = P(n,k)$.

Proof. The lemma can be easily proved by induction on m. □

Lemma 10. $P(n,k,m) \le 2^{n-2}\left(\left(k-\frac{m}{2}\right)^2 + k + \frac{7m}{4} + 8\right) + 2^k\left(\frac{3}{2}\right)^m\left(\frac{k}{2}-\frac{m}{6}\right) - 2^k(k+1) - 2^{n-k}\left(\frac{3}{2}\right)^m$.

Proof. First inequality below is a consequence of Lemma 9 and 6. We also use the following equations: $\sum_{k=0}^{n}\binom{n}{k}x^{k-1}k = n(1+x)^{n-1}$, $\sum_{k=0}^{n}\binom{n}{k}k^2 = n(n+1)2^{n-2}$, $\sum_{k=0}^{n-1}x^{k-1}k = \frac{(1-x)(-nx^{n-1})+(1-x^n)}{(1-x)^2}$.

$$\begin{aligned}
P(n,k,m) &\le \underbrace{\sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}\left((k-j)2^{k-j-1}+2^{n-i-1}\right)}_{(5)} + \underbrace{\sum_{i=0}^{m}\binom{m}{i}B(n-m,k-i)}_{(8)} \\
&= \left(2^k\left(\frac{3}{2}\right)^m\left(k+1-\frac{m}{3}\right) - 2^k(k+1) + m2^{n-1}\right) \\
&\quad + 2^{n-2}\left(k^2 - km + \frac{m(m-1)}{4} + k + 8\right) \\
&\quad + 2^k\left(\frac{3}{2}\right)^m\left(-\frac{k}{2}+\frac{m}{6}-1\right) - 2^{n-k}\left(\frac{3}{2}\right)^m \\
&= 2^{n-2}\left(\left(k-\frac{m}{2}\right)^2 + k + \frac{7m}{4} + 8\right) + 2^k\left(\frac{3}{2}\right)^m\left(\frac{k}{2}-\frac{m}{6}\right) \\
&\quad - 2^k(k+1) - 2^{n-k}\left(\frac{3}{2}\right)^m
\end{aligned}$$

$$\begin{aligned}
&\sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}\left((k-j)2^{k-j-1}+2^{n-i-1}\right) \qquad (5) \\
&\quad = \underbrace{\sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}(k-j)2^{k-j-1}}_{(6)} + \underbrace{\sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}2^{n-i-1}}_{(7)} \\
&\quad = \left(2^k\left(\frac{3}{2}\right)^m(k+1-\frac{m}{3}) - 2^k(k+1)\right) + (m2^{n-1})
\end{aligned}$$

$$\sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}(k-j)2^{k-j-1} = k2^{k-1}\sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}2^{-j} - 2^{k-1}\sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}2^{-j}j \tag{6}$$

$$\begin{aligned}
&= k2^{k-1}\sum_{i=0}^{m-1}\left(\frac{3}{2}\right)^{i} - 2^{k-1}\frac{1}{2}\sum_{i=0}^{m-1}\left(\frac{3}{2}\right)^{i-1} i \\
&= k2^{k-1}2\left(\left(\frac{3}{2}\right)^{m} - 1\right) - 2^{k-1}\left(2 - \left(\frac{3}{2}\right)^{m-1}(3-m)\right) \\
&= 2^{k}\left(\frac{3}{2}\right)^{m}\left(k+1-\frac{m}{3}\right) - 2^{k}(k+1)
\end{aligned}$$

$$\sum_{i=0}^{m-1}\sum_{j=0}^{i}\binom{i}{j}2^{n-i-1} = \sum_{i=0}^{m-1}\left(2^{n-i-1}\sum_{j=0}^{i}\binom{i}{j}\right) = \sum_{i=0}^{m-1}2^{n-i-1}2^{i} = m2^{n-1} \tag{7}$$

$$\begin{aligned}
&\sum_{i=0}^{m}\binom{m}{i}(2^{n-m-2}(k-i)^2 + 2^{n-m-2}(k-i) - 2^{k-i-1}(k-i) \\
&\qquad - 2^{n-m-k+i} + 2^{n-m+1} - 2^{k-i}) \qquad (8)\\
&= \sum_{i=0}^{m}\binom{m}{i}2^{n-m-2}(k-i)^2 + \sum_{i=0}^{m}\binom{m}{i}2^{n-m-2}(k-i) - \sum_{i=0}^{m}\binom{m}{i}2^{k-i-1}(k-i) \\
&\quad - \sum_{i=0}^{m}\binom{m}{i}2^{n-m-k+i} + \sum_{i=0}^{m}\binom{m}{i}2^{n-m+1} - \sum_{i=0}^{m}\binom{m}{i}2^{k-i} \\
&= 2^{n-m-2}(k^2 2^m - km2^m + m(m+1)2^{m-2}) + 2^{n-m-2}(k2^m - m2^{m-1}) \\
&\quad - 2^{k-1}\left(k\left(\frac{3}{2}\right)^{m} - 2^{-m}3^{m-1}m\right) - 2^{n-m-k}3^m + 2^{n+1} - 2^{k}\left(\frac{3}{2}\right)^{m} \\
&= 2^{n-2}\left(k^2 - km + \frac{m(m-1)}{4} + k + 8\right) \\
&\quad + 2^{k}\left(\frac{3}{2}\right)^{m}\left(-\frac{k}{2} + \frac{m}{6} - 1\right) - 2^{n-k}\left(\frac{3}{2}\right)^{m}
\end{aligned}$$

□

Theorem 4. *For* $m = \min(k, n-k)$, $P(n,k) \le 2^{n-2}\left(\left(k - \frac{m}{2} - \frac{7}{4}\right)^2 + \frac{9k}{2} + \frac{79}{16}\right)$ $+2^{k}\left(\frac{3}{2}\right)^{m}\left(\frac{k}{2} - \frac{m}{6}\right) - 2^{k}(k+1) - 2^{n-k}\left(\frac{3}{2}\right)^{m}$.

Proof. Directly from Lemmas 9 and 10. □

We will now present the *size difference* $SD(n,k)$ between pairwise selection network and our network. Merging step in $pw\_sel_K^N$ costs $2^k k - 2^k + 1$ and in $pw\_hbit\_sel_K^N$: $2^{k-1}k$, so the difference is given by the following equation:

$$SD(n,k) = \begin{cases} 0 & \text{if } n = k \\ 0 & \text{if } k = 0 \\ 2^{k-1}k - 2^k + 1 + & \\ +SD(n-1,k) + SD(n-1,k-1) & \text{if } 0 < k < n \end{cases} \tag{9}$$

Theorem 5. *Let* $S_{n,k} = \sum_{j=0}^{k} \binom{n-k+j}{j} 2^{k-j}$. *Then:*

$$SD(n,k) = \binom{n}{k}\frac{n+1}{2} - S_{n,k}\frac{n-2k+1}{2} - 2^k(k-1) - 1$$

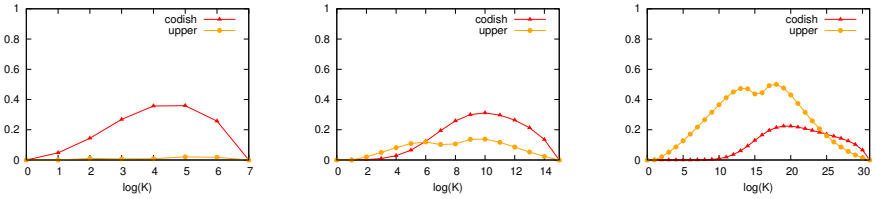
Proof. By straightforward calculation one can verify that $S_{n,0} = 1$, $S_{n,n} = 2^{n+1} - 1, S_{n-1,k-1} = \frac{1}{2}(S_{n,k} - \binom{n}{k})$ and $S_{n-1,k-1} + S_{n-1,k} = S_{n,k}$. It follows that the theorem is true for $k = 0$ and $k = n$. We prove the theorem by induction on pairs (k,n). Take any (k,n), $0 < k < n$, and assume that theorem holds for every $(k',n') \prec (k,n)$ (in lexicographical order). Then we have:

$$\begin{aligned} SD(n,k) &= 2^{k-1}k - 2^k + 1 + SD(n-1,k) + SD(n-1,k-1) \\ &= 2^{k-1}k - 2^k + 1 + \binom{n-1}{k}\frac{n}{2} + \binom{n-1}{k-1}\frac{n}{2} - 2^k(k-1) - 1 \\ &\quad - 2^{k-1}(k-2) - 1 - (S_{n-1,k}\frac{n-2k}{2} + S_{n-1,k-1}\frac{n-2k+2}{2}) \\ &= \binom{n}{k}\frac{n}{2} - S_{n,k}\frac{n-2k}{2} - S_{n-1,k-1} - 2^k(k-1) - 1 \\ &= \binom{n}{k}\frac{n+1}{2} - S_{n,k}\frac{n-2k+1}{2} - 2^k(k-1) - 1 \end{aligned}$$

□

Corollary 1. $|pw\_sel_{N/2}^N| - |pw\_hbit\_sel_{N/2}^N| = N\frac{\log N - 4}{2} + \log N + 2$, *for* $N = 2^n$.

Plots in figure 5 show how much *pw_sel* and the upper bound from Theorem 4 are worse than *pw_hbit_sel*. Lines labeled *codish* are plotted from $(|pw\_sel_K^N| - |pw\_hbit\_sel_K^N|)/|pw\_hbit\_sel_K^N|$ and the ones labeled *upper* are plotted from the formula $(|upper_K^N| - |pw\_hbit\_sel_K^N|)/|pw\_hbit\_sel_K^N|$, where $|upper_K^N|$ is the upper bound from Theorem 4. Both $|pw\_sel_K^N|$ and $|pw\_hbit\_sel_K^N|$ were computed directly from recursive formulas. We can see that we save the most number of comparators when k is larger than $n/2$, nevertheless for small values of n superiority of our network is apparent for any k. As for the upper bound, it gives a good approximation of $|pw\_hbit\_sel_K^N|$ when n is small , but for larger values of n it becomes less satisfactory.



(a) $N = 2^7$ (b) $N = 2^{15}$ (c) $N = 2^{31}$

Fig. 5: Comparison of pairwise selection networks for pratical values of n and k.

6 Arc-consistency of selection networks

In this section we prove that half encoding of any selection network preserves arc-consistency with respect to "less-than" cardinality constraints. The proof can be generalized to other types of cardinality constraints.

We introduce the convention, that $\langle x_1, \ldots, x_n \rangle$ will denote the input and $\langle y_1, \ldots, y_n \rangle$ will denote the output of some order n comparator network. We would also like to view them as sequences of boolean variables, that can be set to either true (1), false (0) or undefined (X).

From now on we assume that every network f is half encoded and when we say "comparator" or "network", we view it in terms of CNF formulas. We denote $V[\phi(f)]$ to be the set of variables in encoding $\phi(f)$.

Observation 4. *A single comparator $hcomp(a, b, c, d)$ has the following* propagation properties*:*

1. *If $a = 1$ or $b = 1$, then UP sets $c = 1$ (by 1.c1 or 1.c2).*
2. *If $a = b = 1$, then UP sets $c = d = 1$ (by 1.c1 and 1.c3).*
3. *If $c = 0$, then UP sets $a = b = 0$ (by 1.c1 and 1.c2).*
4. *If $b = 1$ and $d = 0$, then UP sets $a = 0$ (by 1.c3).*
5. *If $a = 1$ and $d = 0$, then UP sets $b = 0$ (by 1.c3).*

Lemma 11. *Let f_k^n be a selection network. Assume that $k - 1$ inputs are set to 1, and rest of the variables are undefined. Unit propagation will set variables $y_1, \ldots, y_{k-1}$ to 1.*

Proof. From propagation properties of $hcomp(a, b, c, d)$ we can see that if comparator receives two 1s, then it outputs two 1s, when it receives 1 on one input and X on the other, then it outputs 1 on the upper output and X on the lower output. From this we conclude that a single comparator will sort its inputs, as long as one of the inputs is set to 1. No 1 is lost, so they must all reach the outputs. Because the comparators comprise a selection network, the 1s will appear at outputs $y_1, \ldots, y_{k-1}$. □

The process of propagating 1s we call a *forward propagation*. For the remainder of this section assume that: f_k^n is a selection network; $k-1$ inputs are set to 1, and the rest of the variables are undefined; forward propagation has been performed resulting in $y_1, \ldots, y_{k-1}$ to be set to 1.

Definition 13 (path). *A* path *is a sequence of boolean variables* $\langle z_1, \ldots, z_m \rangle$ *such that* $\forall_{1 \leq i \leq m} z_i \in V[\phi(f_k^n)]$ *and for all* $1 \leq i < m$ *there exists a comparator* $hcomp(a, b, c, d)$ *in* $\phi(f_k^n)$ *for which* $z_i \in \{a, b\}$ *and* $z_{i+1} \in \{c, d\}$.

Definition 14 (propagation path). *Let* x *be an undefined input variable. A path* $\bar{z}_x = \langle z_1, \ldots, z_m \rangle$ $(m \geq 1)$ *is a* propagation path*, if* $z_1 \equiv x$ *and* $\langle z_2, \ldots, z_m \rangle$ *is the sequence of variables that would be set to* 1 *by UP, if we would set* $z_1 = 1$.

Lemma 12. *If* $\bar{z}_x = \langle z_1, \ldots, z_m \rangle$ *is a propagation path for an undefined variable* x*, then* $z_m \equiv y_k$.

Proof. Remember that all $y_1, \ldots, y_{k-1}$ are set to 1. Setting any undefined input variable x to 1 will result in UP to set y_k to 1. Otherwise f_k^n would not be a selection network. □

The following lemma shows that propagation paths are deterministic.

Lemma 13. *Let* $\bar{z}_x = \langle z_1, \ldots, z_m \rangle$ *be a propagation path. For each* $1 \leq i \leq m$ *and* $z'_1 \equiv z_i$*, if* $\langle z'_1, \ldots, z'_{m'} \rangle$ *is a path that would be set to* 1 *by UP if we would set* $z'_1 = 1$*, then* $\langle z'_1, \ldots, z'_{m'} \rangle = \langle z_i, \ldots, z_m \rangle$.

Proof. By induction on $l = m - i$. If $l = 0$, then $z'_1 \equiv z_m \equiv y_k$ (by Lemma 12), so the lemma holds. Let $l \geq 0$ and assume that the lemma is true for z_l. Consider $z'_1 \equiv z_{l-1} \equiv z_{m-i-1}$. Set $z_{m-i-1} = 1$ and use UP to set $z_{m-i} = 1$. Notice that $z_{m-i} \equiv z'_2$, otherwise there would exist a comparator $hcomp(a, b, c, d)$, for which z_{m-i-1} is equivalent to either a or b and $z_{m-i} \equiv c$ and $z'_2 \equiv d$ (or vice versa). That would mean that a single 1 on the input produces two 1s on the outputs. This contradicts our reasoning in the proof of Lemma 11. By the induction hypothesis $\langle z'_2, \ldots, z'_{m'} \rangle = \langle z_{m-i}, \ldots, z_m \rangle$, so $\langle z'_1, \ldots, z'_{m'} \rangle = \langle z_{m-i-1}, \ldots, z_m \rangle$. □

For each undefined input variable x and propagation path $\bar{z}_x = \langle z_1, \ldots, z_m \rangle$ we define a directed graph $P_x = \{\langle z_i, z_{i+1} \rangle : 1 \leq i < m\}$.

Lemma 14. *Let* $\{x_{i_1}, \ldots, x_{i_t}\}$ $(t > 0)$ *be the set of undefined input variables. Then* $T = P_{x_{i_1}} \cup \ldots \cup P_{x_{i_t}}$ *is the tree rooted at* y_k.

Proof. By induction on t. If $t = 1$, then $T = P_{x_{i_1}}$ and by Lemma 12, $P_{x_{i_1}}$ ends in y_k, so the lemma holds. Let $t > 0$ and assume that the lemma is true for t. We will show that it is true for $t + 1$. Consider $T = P_{x_{i_1}} \cup \ldots \cup P_{x_{i_t}} \cup P_{x_{i_{t+1}}}$. By the induction hypothesis $T' = P_{x_{i_1}} \cup \ldots \cup P_{x_{i_t}}$ is a tree rooted at y_k. By Lemma 12, $V(P_{x_{i_{t+1}}}) \cap V(T') \neq \emptyset$. Let $z \in V(P_{x_{i_{t+1}}})$ be the first variable, such that $z \in V(T')$. Since $z \in V(T')$, there exists j $(1 \leq j \leq t)$ such that $z \in P_{x_{i_j}}$. By Lemma 13, starting from variable z, paths $P_{x_{i_{t+1}}}$ and $P_{x_{i_j}}$ are identical. □

Graph T from the above lemma will be called a *propagation tree*.

Theorem 6. *If we set $y_k = 0$, then unit propagation will set all undefined input variables to* 0.

Proof. Let T be the propagation tree rooted at y_k. We prove by induction on the height h of T, that (*) if we set root of T to 0, then all nodes of the tree will be set to 0, thus all undefined input variables will also be set to 0. If $h = 0$, then $V = \{y_k\}$, so (*) is trivially true. Let $h > 0$ and assume that (*) holds. We will show that (*) holds for height $h + 1$. Let T' be the propagation tree of height $h + 1$ and let $r = 0$ be the root. Consider children of r in T' and a comparator $hcomp(a, b, c, d)$ for which $r \in \{c, d\}$:

Case 1: r has two children. The only case is when $r \equiv c = 0$. Unit propagation sets $a = b = 0$. Nodes a and b are roots of propagation trees of height h and are set to 0, therefore by the induction hypothesis all nodes in T' will be set to 0.

Case 2: r has one child. Consider two cases: (i) if $r \equiv c = 0$ and either a or b is the child of r, then UP sets $a = b = 0$ and either a or b is the root of propagation tree of height h and is set to 0, therefore by the induction hypothesis all nodes in T' will be set to 0, (ii) $r \equiv d = 0$ and either $a = c = 1$ and b is the child of r or $b = c = 1$ and a is the child of r. Both of them will be set to 0 by UP and again we get the root of propagation tree of height h that is set to 0, therefore by the induction hypothesis all nodes in T' will be set to 0. □

7 Conclusions

We have constructed a new family of selection networks, which are based on the pairwise selection ones, but require less comparators to merge subsequences. The difference in sizes grows with k and is equal to $n\frac{\log n-4}{2} + \log n + 2$ for $k = n/2$. In addition, we have shown that any selection network encoded in a standard way to a CNF formula preserves arc-consistency with respect to a corresponding cardinality constraint. This property is important, as many SAT-solvers take advantage of arc-consistency, making the computation significantly faster.

It's also worth noting that using encodings based on selection networks give an extra edge in solving optimization problems for which we need to solve a sequence of problems that differ only in the decreasing bound of a cardinality constraint. In this setting we only need to add one more clause $\neg y_k$ for a new value of k, and the search can be resumed keeping all previous clauses as it is. This works because if a comparator network is a k-selection network, then it is also a k'-selection network for any $k' < k$. This property is called *incremental strengthening* and most state-of-the-art SAT-solvers provide a user interface for doing this.

References

1. Asín R., Nieuwenhuis R., Oliveras A., Rodríguez-Carbonell E. (2009). Cardinality networks and their applications. *SAT*, pp. 167–180.

2. Asín R., Nieuwenhuis R., Oliveras A., Rodríguez-Carbonell E. (2011). Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221.
3. Batcher K.E. (1968). Sorting networks and their applications. *AFIPS Spring Joint Computing Conference*, pp. 307–314.
4. Codish M., Zazon-Ivry M. (2010). Pairwise Cardinality Networks. *LPAR, LNCS volume 6355, Springer*, pp. 154–172.
5. Codish M., Zazon-Ivry M. (2012). Pairwise Networks are Superior for Selection. Manuscript: `http://www.cs.bgu.ac.il/~mcodish/Papers/Sources/pairwiseSelection.pdf`.
6. Eén N., Sorensson N. (2006). Translating pseudo-boolean constraints into sat. *JSAT*, 2:1–26.
7. Knuth D.E. (1973). The Art of Computer Programming, Volume III: Sorting and Searching. *Addison-Wesley*.
8. Parberry I. (1987). Parallel complexity theory. *Research notes in theoretical computer science. Pitman*.
9. Parberry I. (1992). The pairwise sorting network. *Parallel Processing Letters*, 2:205–211.